TRANSMITTAL OF APPEAL BRIEF

Docket No.
M060

In re Application of: Charles A. McBrian et al.

Application No.	Filing Date	Examiner	Group Art Unit
10/688,062-Conf. #8145	October 17, 2003	J. R. Swearingen	2145

Invention: LIVE-SERVER CONTENT STAGING

TO THE COMMISSIONER OF PATENTS:

Transmitted herewith is the Appeal Brief in this application, with respect to the Notice of Appeal filed: May 27, 2008 .

The fee for filing this Appeal Brief is $ 510.00 .

[x] Large Entity [ ] Small Entity

[ ] A petition for extension of time is also enclosed.

The fee for the extension of time is _____ .

[ ] A check in the amount of _____ is enclosed.

[ ] Charge the amount of the fee to Deposit Account No. 06-2380 .
This sheet is submitted in duplicate.

[x] Payment being paid on-line by credit card.

[x] The Director is hereby authorized to charge any additional fees that may be required or credit any overpayment to Deposit Account No. 06-2380 .

Dated: July 31, 2008

Thomas J. Meaney
Attorney Reg. No. : 41,990
FULBRIGHT & JAWORSKI L.L.P.
2200 Ross Avenue, Suite 2800
Dallas, Texas 75201-2784
(214) 855-8278

Appeal Brief Transmittal

I hereby certify that this paper (along with any paper referred to as being attached or enclosed) is being transmitted via the Office electronic filing system in accordance with § 1.6(a)(4).

Dated: July 31, 2008 Signature: Donna Forbit (Donna Forbit)

Appeal Brief
I hereby certify that this paper (along with any paper referred to as being attached or enclosed) is being transmitted via the Office electronic filing system in accordance with § 1.6(a)(4).
Dated: July 31, 2008 Signature: Donna Forbit (Donna Forbit)

Docket No.: M060
(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
Charles A. McBrian

Application No.: 10/688,062 Confirmation No.: 8145

Filed: October 17, 2003 Art Unit: 2145

For: LIVE-SERVER CONTENT STAGING Examiner: J.R. Swearingen

APPEAL BRIEF

MS Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

As required under 37 C.F.R. § 41.37(a), this brief is filed within two months of the Notice of Appeal filed in this case on May 27, 2008, and is in furtherance of said Notice of Appeal.

The fees required under 37 C.F.R. § 41.20(b)(2) are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF.

This brief contains items under the following headings as required by 37 C.F.R. § 41.37 and M.P.E.P. § 1206:

I.	Real Party In Interest
II	Related Appeals and Interferences
III.	Status of Claims
IV.	Status of Amendments
V.	Summary of Claimed Subject Matter
VI.	Grounds of Rejection to be Reviewed on Appeal
VII.	Argument
VIII.	Claims Appendix
IX.	Evidence Appendix
X.	Related Proceedings Appendix

I. REAL PARTY IN INTEREST

The real party in interest for this appeal is:

ADOBE SYSTEMS INCORPORATED

II. RELATED APPEALS, INTERFERENCES, AND JUDICIAL PROCEEDINGS

There are no other appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

A. Total Number of Claims in Application

There are 20 claims pending in application.

B. Current Status of Claims

1. Claims canceled: None
2. Claims withdrawn from consideration but not canceled: None
3. Claims pending: 1-20
4. Claims allowed: None
5. Claims rejected: 1-20

C. Claims On Appeal

The claims on appeal are claims 1-20

IV. STATUS OF AMENDMENTS

Applicant filed an Amendment After Final Rejection on March 10, 2008. The Examiner maintained the rejections of the claims in an Advisory Action mailed April 11, 2008. Applicant did not make any amendments in the After Final Amendment. Therefore, the claims on appeal are those as rejected in the Final Office Action of February 25, 2008. A complete listing of the claims is provided in the Claims Appendix attached hereto.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The following provides a concise explanation of the subject matter defined in each of the separately argued claims involved in the appeal, referring to the specification by page and line number and to the drawings by reference characters, as required by 37 C.F.R. § 41.37(c)(1)(v). Each element of the claims is identified by a corresponding reference to the specification and drawings where applicable. It should be noted that the citation to passages in the specification and drawings for each claim element does not imply that the limitations from the specification and drawings should be read into the corresponding claim element.

According to one claimed embodiment, such as that of independent claim 1, a method for staging file assets on a live server (*e.g.,* live Web server 103 of Figures 2-3) includes detecting an index page (*e.g.,* index.html 302 of Figure 3A) of the server (*e.g.,* Step 400 of Figure 4, and ¶¶ 17, 18, 21), creating a staging folder (*e.g.,* staging folder 303 of Figure 3A) within a file system of the server, wherein the staging folder does not default to a directory listing of the file system when accessed (*e.g.,* ¶¶ 17-18), inserting a randomized string (*e.g.,* randomized string 306 of Figure 3B) into a name of the file assets to be staged (step 403 of Figure 4, and ¶ 21), and storing the file assets in the staging folder (step 403 of Figure 4, and ¶ 21).

In certain embodiments, such as that of claim 2, the method of claim 1 further includes communicating the name with the inserted randomized string to a reviewing party (*e.g.,* ¶ 19).

In certain embodiments, such as that of claim 3, the detecting part of claim 1 includes creating a temporary folder to a root directory of the live server (*e.g.*, live Web server of Figure 2-3, step 500 of Figure 5, and ¶ 22), writing a page of content to the temporary folder according to one of a plurality of standard addresses (*e.g.,* step 501 of Figure 5 and ¶ 22), attempting to access the temporary folder over hypertext transfer protocol (HTTP) from the server (*e.g.,* step 502 of Figure 5, and ¶ 22), writing the page of content to the temporary folder using another of the plurality of standard addresses when the file cannot be read in the attempting step (*e.g.,* step 504 of Figure 5, and ¶ 22), and validating the one of the plurality of standard addresses when the index page is read back in the attempting step (*e.g.,* step 503 of Figure 5, and ¶ 22).

In certain embodiments, such as that of claim 4, the plurality of standard addresses from claim 3 is stored on a memory accessible by a development environment (*e.g.*, ¶ 22).

In certain embodiments, such as that of claim 5, the creating part of claim 1 further includes generating a blank index file (*e.g.*, blank index 304 of Figure 3A, and ¶ 18) named according to the detected index file (*e.g.*, step 402 of Figure 4, and ¶ 18) and storing the blank index file in the staging folder (*e.g.*, step 402 of Figure 4, and ¶ 18), wherein the blank index file inhibits default directory listing of the staging folder in the file system (*e.g.*, ¶ 18).

In certain embodiments, such as that of claim 6, the method described in claim 1 further includes generating the randomized string (*e.g.*, randomized string 306 of Figure 3B) prior to the inserting step (*e.g.*, ¶¶ 19-20).

In certain embodiments, such as that of claim 7, the method of claim 1 further includes determining a length of the randomized string(*e.g.*, randomized string 306 of Figure 3B), wherein the length corresponds to a desired level of security (*e.g.*, ¶ 19).

According to one claimed embodiment, such as that of independent claim 8, a computer program product is defined having a computer readable medium with computer program logic recorded thereon. The computer program logic facilitates staging file assets. The computer program logic includes code that, when executed by a computer (*e.g.*, live Web server 103) causes, the computer to perform a method including detecting, in a file system of a Web server, an index (*e.g.*, index file 302 of Figure 3A) of the Web server (*e.g.*, step 400 of Figure 4), generating, in the file system of the Web server, a staging folder (*e.g.*, staging folder 303 of Figure 3A, step 401 of Figure 4, and ¶¶ 18, 20), storing, in the staging folder, the file assets to be staged according to names that include a random string (*e.g.*, randomized string 305 of Figure 3B, step 403 of Figure 4, and ¶¶ 19-20), wherein the file assets are served by the Web server to users accessing the staging folder (*e.g.*, ¶¶ 19-20), and inhibiting listing of the staging folder in a default directory listing of the file system by the Web server (*e.g.*, ¶¶ 17-18).

In certain embodiments, such as that of claim 9, the detecting part of the computer program product of claim 8 includes creating a temporary folder in root directory (*e.g.*, step 500 of Figure 5, and ¶ 22), writing a test file to the root directory using one of a plurality of

standard index locations (*e.g.,* step 501 of Figure 5, and ¶ 22), requesting access to the test file from the Web server (*e.g.,* step 502 of Figure 5, and ¶ 22), writing the test file to the temporary folder using another of the plurality of standard index locations when the test file cannot be accessed during the requesting (*e.g.,* step 501 of Figure 5, and ¶ 22), and certifying the one of the plurality of standard index locations when the test file is accessed during the requesting (*e.g.,* step 503 of Figure 5, and ¶ 22).

In certain embodiments, such as that of claim 10, the plurality of standard addresses of the computer program product of claim 9 is stored on a memory accessible by a development environment (*e.g.,* ¶ 22).

In certain embodiments, such as that of claim 11, the generating part of the computer program product of claim 8 further includes creating an empty index named according to the detected index (*e.g.,* blank index 304 of Figure 3A, and ¶ 18) and storing the empty index in the staging folder (*e.g.,* step 402 of Figure 4, and ¶ 18), wherein the empty index inhibits the listing of the staging folder in the default directory listing of the file system by the Web server (*e.g.,* ¶ 18).

In certain embodiments, such as that of claim 12, the computer program product of claim 8 further includes generating the random string (*e.g.,* randomized string 306 of Figure 3B) prior to the storing (*e.g.,* ¶¶ 19-20).

In certain embodiments, such as that of claim 13, the length of the random string of the computer program product of claim 8 is determined, wherein the length corresponds to a desired level of security (*e.g.,* ¶ 19).

According to one claimed embodiment, such as that of independent claim 14, a method for reviewing proposed file content on a live Web server (*e.g.,* live Web server 103 of Figures 2-3) includes scanning the live Web server for an index file (*e.g.,* index file 302 of Figure 3, step 400 of Figure 4, and ¶ 18), opening a review folder (*e.g.,* staging file 303 of Figure 3A) in a file system of the live Web server (*e.g.,* ¶ 18), creating a blank index (*e.g.,* blank index file 304 of Figure 3A) on the review folder (*e.g.,* ¶ 18), wherein the blank index is named according to a name of the index file (*e.g.,* ¶ 18), wherein the blank index inhibits listing by the Web server of the review folder in a default directory listing of the file system

(*e.g.*, ¶¶ 17-18), and storing the proposed file content in the review folder (step 403 of Figure 4, and ¶¶ 17-18), wherein a randomized string (*e.g.*, randomized string 306 of Figure 3B) is included in a file name representing the proposed file content (*e.g.*, , step 403 of Figure 4, and ¶¶ 19-20), and wherein the proposed file content is served by the Web server to users accessing the file name (*e.g.*, ¶ 19).

In certain embodiments, such as that of claim 15, the method of claim 14 further includes sharing the file name with a reviewing user (*e.g.*, ¶ 19).

In certain embodiments, such as that of claim 16, the scanning part of the method of claim 14 includes creating a temporary folder on the web root of the Web server (*e.g.*, step 500 of Figure 5, and ¶ 22), retrieving a first index address from a plurality of standard index addresses (*e.g.*, step 501 of Figure 5, and ¶ 22), writing a test document to the temporary folder using the first index address (*e.g.*, step 501 of Figure 5, and ¶ 22), requesting the test document from the live Web server over hypertext transfer protocol (HTTP) (*e.g.*, step 502 of Figure 5, and ¶ 22), and marking the first index address valid when the test document is retrieved during the requesting step (*e.g.*, step 503 of Figure 5, and ¶ 22).

In certain embodiments, such as that of claim 17, the method of claim 16 further includes retrieving a next index address from the plurality of standard index addresses when the test document is not retrieved during the requesting step (*e.g.*, step 504 of Figure 5, and ¶ 22), writing the test document to the temporary folder using the next index address (*e.g.*, step 501 of Figure 5, and ¶ 22), requesting the next test document from the live Web server (*e.g.*, step 502 of Figure 5, and ¶ 22), marking the next index address valid when the test document is retrieved during the requesting the next index step (*e.g.*, step 503 of Figure 5, and ¶ 22), and repeating from the retrieving the next index step when the next test document is not retrieved during the requesting the next index step (*e.g.*, step 504 of Figure 5, and ¶ 22).

In certain embodiments, such as that of claim 18, the method of claim 14 further includes determining a level of security for the proposed file content (*e.g.*, ¶ 19), establishing a number of characters for the randomized string (*e.g.*, randomized string 306 of Figure 3B) responsive to the determined level of security (*e.g.*, ¶ 19), and generating a random character for each of the number of established characters (*e.g.*, ¶ 19).

In certain embodiments, such as that of claim 19, the method of claim 1 further includes the server serving the file assets to users who access the staging folder (*e.g.*, staging folder 303 of Figure 3A, and ¶ 19).

In certain embodiments, such as that of claim 20, the staging folder (*e.g.*, staging folder 303 of Figure 3A)of the method of claim 19 is not discoverable through a default directory listing of the file system by the server (*e.g.*, ¶¶ 17-18).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1-20 stand rejected under 35 U.S.C. §102(e) as being anticipated by U.S. Patent No. 6,792,454 to Nakano, et al., (hereinafter *Nakano*).

VII. ARGUMENT

Appellant respectfully traverses the outstanding rejections of the pending claims, and requests that the Board reverse the outstanding rejections in light of the remarks contained herein. The claims do not stand or fall together. Instead, Appellant presents separate arguments for various independent and dependent claims. Each of these arguments is separately argued below and presented with separate headings and sub-heading as required by 37 C.F.R. § 41.37(c)(1)(vii).

The current claims 1-20 each stand rejected under 35 U.S.C. § 102(e) as anticipated by *Nakano*. "A claim is anticipated ***only if*** each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987) (emphasis added). Moreover, "[t]he ***identical invention*** must be shown in as complete detail as is contained in the . . . claim." *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236 (Fed. Cir. 1989) (emphasis added). Therefore, in order to uphold the Examiner's rejections of claims 1-20, the Board must find the ***identical invention*** in *Nakano* shown in as complete detail as contained in claims 1-20. While Applicant need only show the absence of a single element in the independent claims in order to overcome the § 102(e) rejection, Applicant asserts that *Nakano* may, in fact, only disclose a single element out of all the elements found in claims 1-20.

A. Preamble Limitations

The preamble of claim 1 provides, "A method for staging file assets on a live server comprising ...." The preamble of claim 14 provides, "A method for reviewing proposed file content on a live Web server comprising ...." One of the critical differences between the claimed invention and *Nakano* is that the claimed invention allows staging file assets or content that is destined to make it onto a Web page directly on the live Web server or live server on which the Web page is accessible by all Internet users. ¶¶ [0004]-[0007] and [0017]-[0023]. *Nakano*, on the other hand, teaches the prior art method of having a separate staging server (development server 130) from the live server or live Web server (website production server 170). *Nakano*, col. 5, lns 5-23; Figure 1. This prior art configuration is even discussed in the Specification at ¶ [0016].

In rejecting Applicant's argument regarding the live server, the Examiner stated that "the recitation, 'on a live server' has not been given patentable weight because the recitation occurs in the preamble." Advisory Action, Continuation Sheet. The Examiner continues that:

> A preamble is generally not accorded any patentable weight where it merely recites the purpose of a process or the intended use of a structure, and where the body of the claim does not depend on the preamble for completeness but, instead, the process steps or structural limitations are able to stand alone. Advisory Action, Continuation Sheet (*citing In re Hirao*, 535 F.2d 67 (CCPA 1976) and *Kropa v. Robie*, 187 F.2d 150, 152 (CCPA 1951).)

The Examiner correctly cites to the case law that holds that, in general, the preamble bears no patentable weight. However, more recent Federal Circuit case law also holds that:

> If the claim preamble, when read in the context of the entire claim, recites limitations of the claim, or, if the claim preamble is 'necessary to give life, meaning, and vitality' to the claim, then the claim preamble should be construed as if in the balance of the claim. *Pitney Bowes, Inc., v. Hewlett-Packard Co.,* 51 U.S.P.Q.2d 1161, 1165-66 (Fed. Cir. 1999).

Claim 1 begins with the preamble identifying the method for staging files "on a live server." Claim 1 continues, "detecting an index page of ***said server*** ...." Claim 1 further requires, "creating a staging folder within a file system of ***said server*** ...." Moreover, this

staging file that is created within a file system of the live server is again addressed in "storing said file assets in said staging folder." According to the analogy of the Examiner's own recitation of the case law, the preamble ***is*** given patentable weight where the body of the claim depends on the preamble for completeness and the body cannot stand alone. *See* Advisory Action, Continuation Sheet. With regard to claims 1 and 14, it is clear that the bodies of the claims integrally rely on the preambles' first mention of the "live server" and "live Web server." Not only is an index page detected from this live server, but a staging/review folder is created on the live server, into which file assets/content are stored. Without the first use of the terms "live server" and "live Web server," the bodies of claims 1 and 14 could not stand alone. Therefore, having the first referenced instance of the "live server" and "live Web server" in the preambles does not render the term patentably meaningless. Accordingly, Applicant respectfully requests the Board to overrule the Examiner's rejections of at least claims 1-7 and 14-20 because the asserted *Nakano* reference simply does not teach a staging folder on the live server.

B. Independent Claim 1

Claim 1 requires, "... detecting an index page of said server ...." The Examiner asserts that this limitation is taught by *Nakano* at column 22, lines 21-29. Final Office Action, p. 2. Column 22, lines 21-29 of *Nakano* provides:

> An example of a mapping rule for the document root of a particular branch is as follows:
>
> _docroot=/directoryA/directoryB
>
> If the original URL request is GET /documents/idex.html, the document root prefix will be appended directly in front of the first "/" of the original request. In other words, the prefix "/directoryA/directoryB" will be appended to the front of "/documents/index.html".

Nothing within this cited selection from *Nakano*, or any other part of *Nakano*, discusses detecting an index page. The selection gives an example URL request that could be used with a mapping rule for appending a document root to the resulting pathway. Col. 22, lns 21-29. The example URL request includes "index.html." However, index pages are very well known in the art. The fact that an example URL contains "index.html" does not mean that

the index page is being detected for the live server, as required by claim 1. Accordingly, *Nakano* does not teach this limitation of claim 1.

Claim 1 also requires, "... creating a staging folder within a file system of said server, wherein said staging folder does not default to a directory listing of said file system when accessed ...." The Examiner asserts that this limitation is disclosed by *Nakano* at column 2, lines 48-52. Final Office Action, p. 2. Column 2, lines 48-52 of *Nakano* states:

> A staging area is a read-only file system that supports select versioning operations. Various users of work areas can integrate their work by submitting the contents of their work areas to the staging area. In the staging area, developers can compare their work and see how their changes fit together.

This cited selection provides for multiple users submitting their work content to the staging area to compare and work on the changes. It also defines what a staging area is. However, nothing within this selection, or any other part of *Nakano*, teaches that a staging folder is created in the live server nor that the staging folder does not default to a directory listing of the file system when accessed. Thus, *Nakano* does not teach this limitation of claim 1.

The Examiner adds, in his comments of the Advisory Action, that *Nakano* teaches that its staging folder does not default to a directory listing of the file system when accessed. Advisory Action, Continuation Sheet. The Examiner contends, "As shown in column 5, lines 15-23, the development server is coupled to a production server, which can be read as the default directory listing and the entire production server as the staging folder/area." Advisory Action, Continuation Sheet. Applicant does not even understand the meaning of this statement. The entire production server, which is the live server containing the files for the live Web page, cannot be construed as the staging ***folder***, because the production server, as used in *Nakano,* does not even meet the definition of what a staging folder/area is (i.e., a folder in which proposed content is located prior to finalizing and uploading to the live Web server which is not easily accessible by the general public), not to mention that it is difficult to reasonably support that an entire server may be construed as a folder.

Alternatively, the Examiner offers that *Nakano*'s disclosure of copying the staging areas into edition areas to create an edition of the Website somehow also teaches that the staging folder does not default to a directory listing of the file system when accessed, as

required by claim 1. Advisory Action, Continuation Sheet. However, the copying of the staging areas into an edition area, as described in column 6, lines 52-58, does not logically equate to a staging folder that does not default to a directory listing of the file system when accessed. Nothing in *Nakano* teaches or even discusses directory listings of file systems, let alone a staging folder that does not default to such a directory listing, as required by claim 1. Thus, *Nakano* does not teach this limitation.

Claim 1 also requires, "... inserting a randomized string into a name of said file assets to be staged ...." The Examiner contends that *Nakano* teaches this limitation at column 8, lines 18-29 and lines 52-62. Final Office Action, p. 2. The cited selections from *Nakano* provide:

> Each work area, staging area, and edition area has two unique identifiers, one of which is referred to in this application as a "generation ID," and the other of which is referred to as an "object ID." The object ID identifies the object that represents the area, and, once an object ID is assigned to an object, that object ID is not changed. Each area is also identified by a unique generation ID, which indicates how an area is related to other areas. The generation ID for a particular area can be changed, as will be discussed below (e.g., when a staging area is published into an edition). The generation ID is placed in the generation ID field. The object ID is placed in the object ID field. Col. 8, lns 18-29.
>
> FIG. 10 illustrates the method for deriving a generation ID, which is to be assigned to a new area, from a parent generation ID, where the parent generation ID is assigned to the direct parent of the new area. A unique number is obtained 1000 using a conventional algorithm for sequentially (e.g., 1, 2, 3, 4) or randomly generating unique numbers. The set or sequence of unique numbers associated with the parent generation ID is then retrieved 1010. Subsequently, a set or sequence of numbers that is the concatenation of the parent generation ID and the just issued unique number is created 1020. Col. 8, lns 52-62.

Lines 18-29 certainly only discuss the generation ID and the object ID that are used to uniquely identify each work area, staging area, and edition area, and nothing regarding a randomized string. Lines 52-62 discuss creating a new parent generation ID by adding either a sequential number or a randomly generated number to the existing new parent generation

ID. However, the mere fact that this selection uses the word "randomly" does not mean that it teaches this limitation of claim 1. While a randomly generated number is added to create a new parent generation ID, this new ID refers to the particular area, whether it's a work area, a staging area, or an edition area. It is not a name of a file asset that is intended to be staged, as required by claim 1. Therefore, while *Nakano* uses the word "randomly," it does not teach inserting a randomized string into a name of file assets to be staged. Thus, *Nakano* does not teach this limitation of claim 1.

Claim 1 further requires, "... storing said file assets in said staging folder." The Examiner contends that *Nakano* teaches this limitation at column 6, lines 40-58. Final Office Action, p. 2. *Nakano* column 6, lines 40-58 provide:

> The collective work in a staging area can be virtually copied back to the private work areas to keep the work areas up-to-date with the current state of the staging areas, which changes as different contributors submit new content from work areas. The copying is "virtual" because areas share directory trees so that the directory trees do not have to be physically copied. When the collective work in a staging area is deemed final, its contents can be published to create an edition of the website. Creating a work area from an edition and publishing a staging area are additional ones of the select versioning operations discussed below.
>
> The contents of a staging area are virtually copied into an edition area to create an edition of a website. Again, virtually copying means that the edition references the same directory tree as the staging area. Because an edition is a read-only file system, it is a frozen snapshot of the content of the entire website at a particular point along a single branch. Each edition is archived and accessible to all developers.

This cited selection from *Nakano* does not even teach that file assets are stored into a staging folder. It discusses what happens to files or information that are already in the staging folder. Col. 6, lns 40-44. Moreover, because *Nakano* does not teach that the staging file is on the live server, *Nakano* cannot teach this limitation of claim 1. Accordingly, for each of the above reasons, Applicant respectfully requests the Board to overrule the Examiner's rejection of claim 1.

C. Claim 2

Claim 2 requires, "... communicating said name with said inserted randomized string to a reviewing party." The Examiner asserts that *Nakano* teaches this limitation at column 6, lines 26-40. Final Office Action, p. 2. Lines 26-40 provide:

> Developers integrate their work in a staging area by submitting the contents of their work areas into a staging area. The submit operation is one of the select versioning operations referenced above and discussed in more detail below. The staging area is a shared view of the website available to all users on a branch (branches are discussed below). In other words, a staging area is a file system that is accessible to all users along a branch. A staging area holds the collective work of several developers' work areas and allows the developers to share and integrate their changes. In a staging area, the developers can compare their work and see how their changes fit together. The compare operation is another one of the select versioning operations discussed in more detail below.

However, this selection from *Nakano* discusses only that developers can integrate their work by uploading it all to the staging area within which the developers may collaborate on further modifications. Col. 6, lns 26-36. There is nothing in this cited selection or any other disclosure of *Nakano* that even suggests communicating the new name with the randomized string inserted into it to a reviewing party. Thus, *Nakano* does not teach the limitations of claim 2 even regardless of claim 2's dependence from claim 1.

D. Claim 3

Claim 3 requires, "... creating a temporary folder to a root directory of said live server ...." The Examiner contends that *Nakano* teaches this limitation at column 7, lines 1-7. Final Office Action, pp. 2-3. Column 7, lines 1-7 provide:

> ... the staging area. In the staging area, the contents of the contributor's work area is integrated 230 with the contents of other contributors' work areas. If all the contents submitted to the staging area integrate well and are approved 240, the contents of the staging area are published 250 into an edition. Otherwise, the contributors continue to edit 210 their files until they are approved.

This cited selection of *Nakano* discusses submission of the contents of the individual contributors to the staging area where they are edited, finalized, and then published as an edition. It does not discuss anything with regard to creating a temporary folder to a root directory of the live server, as required by claim 1. Thus, *Nakano* does not teach this limitation of claim 3.

Claim 3 also requires, "... writing a page of content to said temporary folder according to one of a plurality of standard addresses ...." The Examiner contends that *Nakano* teaches this limitation at the same column 7, lines 1-7, as reproduced above. Final Office Action, p. 3. Again, however, neither this selection, nor any other part of *Nakano* teach or suggest that a page of content is written to the temporary folder according to one of many standard addresses. This selection discusses moving content from contributors' individual work areas to the staging area. *Nakano* does not teach that the staging area is a temporary folder that is created on the live server. There is also no discussion of using one or many standard addresses in *Nakano*. Thus, *Nakano* does not teach this limitation of claim 3.

Claim 3 also requires, "... attempting to access said temporary folder over hypertext transfer protocol (HTTP) from said server ...." Again, the only server claimed in claims 1 and 3 is the live server. Therefore, the attempt at accessing is coming from the live server. The Examiner contends that *Nakano* teaches this limitation at column 5, line 11, which states, "Also stored in the memory 150 is a HTTP protocol virtualization module 137 which, as discussed below, the processor 140 executes to allow web server 145 to operate as if it were multiple web severs." Col. 5, lns 10-14. This selection from *Nakano* uses the phrase "HTTP protocol." However, as noted above, use of one of the same terms found in the claim does not mean that the claim is anticipated. HTTP protocol is the communication protocol of web pages. *Nakano* does not teach trying to access a temporary folder from the live server over HTTP. Thus, *Nakano* does not teach this limitation of claim 3.

Claim 3 further requires, "... writing said page of content to said temporary folder using another of said plurality of standard addresses when said file cannot be read in said attempting step ...." Claim 3 describes the function of how this process can detect the index page of the live server. The claim 3 process uses a trial and error process in which known standard address names for these index pages are used in a write-and-read process. If the

page of content is successfully read when the process attempts access, then the system knows it has found the correct standard address of the index page. This limitation is the "retry" using another of the standard addresses when the first attempted access failed. The Examiner believes that this limitation is found in column 6, lines 51-58 of *Nakano*. Final Office Action, p. 3. Lines 51-58 provide:

> The contents of a staging area are virtually copied into an edition area to create an edition of a website. Again, virtually copying means that the edition references the same directory tree as the staging area. Because an edition is a read-only file system, it is a frozen snapshot of the content of the entire website at a particular point along a single branch. Each edition is archived and accessible to all developers.

This selection only discusses copying the contents of the staging area into the edition area to create an actual edition of the web site. It has no contextual bearing on the limitations presented in claim 3. As noted previously, *Nakano* does not discuss attempting various standard addresses. This is because *Nakano* teaches nothing similar to the claimed invention. It further does not teach or even suggest that any writing occurs when the file cannot be read in the previous attempting step. Thus, *Nakano* does not teach this limitation of claim 3.

Claim 3 further requires, "... validating said one or said plurality of standard addresses when said index page is read back in said attempting step." Here, the process of claim 3 has found the correct address of the list of standard addresses that it uses, and may, thus, validate the correct address. The Examiner contends that *Nakano* teaches this limitation at the same column 7, lines 1-7 noted above. Final Office Action, p. 3. However, the Board can clearly see from this selection that there is no such validating of a standard address used as index pages that occurs in either this selection or any of the teachings of *Nakano*. Thus, *Nakano* does not teach this limitation of claim 3. Accordingly, Applicant respectfully requests the Examiner to overrule the Examiner's rejection of claim 3.

E. Claim 4

Claim 4 requires, "... wherein said plurality of standard addresses is stored on a memory accessible by a development environment." The Examiner contends that *Nakano*

teaches this limitation at column 8, lines 18-37. Final Office Action, p. 3. Column 8, lines 18-37, part of which was provided above, state:

> Each work area, staging area, and edition area has two unique identifiers, one of which is referred to in this application as a "generation ID," and the other of which is referred to as an "object ID." The object ID identifies the object that represents the area, and, once an object ID is assigned to an object, that object ID is not changed. Each area is also identified by a unique generation ID, which indicates how an area is related to other areas. The generation ID for a particular area can be changed, as will be discussed below (e.g., when a staging area is published into an edition). The generation ID is placed in the generation ID field. The object ID is placed in the object ID field.
>
> Note that directories, files, history objects (discussed below), and other objects are also assigned object IDs which remain unchanged once assigned.
>
> A generation ID is comprised of a unique sequence or set of numbers. A generation ID not only uniquely identifies an area, but, as stated above, also indicates how an area is related to other areas.

However, this selection from *Nakano* describes only the various identifiers for the work areas, staging area, and edition areas. Each such area has two unique identifiers: the generation ID and the object ID. Col. 8, lns 18-21. These various IDs that identify particular areas of the *Nakano* invention are not the same as a plurality of standard addresses as used in claim 4. Thus, *Nakano* does not teach this limitation. Accordingly, Application respectfully requests the Examiner to overrule the Examiner's rejection of claim 4.

F. Claim 5

Claim 5 requires that the creating step of claim 1 further requires, "... generating a blank index file named according to said detected index file ...." The Examiner contends that *Nakano* teaches this limitation at column 11, lines 6-14. Final Office Action, p. 3. Lines 6-14 provide:

> The module 1314 creates 1530 an object that represents the staging area and that has a name field, a root directory field, a generation ID field, an object ID field, and a branch field. The module 1314 fills 1540 in the name of the staging area in the

> name field, the root directory of edition e in the root directory field, the generation ID obtained in step 1520 in the generation ID field, the object ID obtained in step 1525 in the object ID field, and the identification of the branch along which the staging area lies in the branch field.

This cited selection discusses creating an object representing the staging area that has all of the appropriate IDs and fields associated with the staging area. It does not teach that a blank file is created that is then named according to the index file that was detected, as required in claim 5. Thus, *Nakano* does not teach this limitation of claim 5.

Claim 5 further requires that the creating step requires, "... storing said blank index file in said staging folder, wherein said blank index file inhibits default directory listing of said staging folder in said file system." The Examiner contends that *Nakano* teaches this limitation at column 11, lines 16-62. Final Office Action, p. 3. Column 11, lines 16-62 provide:

> It is sometimes desirable to have multiple staging areas. For instance, in addition to a public staging area it may be desirable to have a few private staging areas to which only select users can submit content. Multiple staging areas can be created by performing the above-described staging area creation method multiple times.
>
> FIGS. 16a-b illustrate the operation of the edition creation module 1316 for creating an edition from the contents of a staging area, arbitrarily labeled "s." The module 1316 obtains 1610 the generation ID and the root directory of staging area s. The module also obtains 1615 a unique object ID for edition e. The module 1316 creates 1620 an object that represents the new edition and that has a name field, a root directory field, a generation ID field, an object ID field, and a branch field. The module 1316 fills 1630 in the name field with the name of the new edition, the generation ID field with the generation ID of staging area s, the root directory field with the root directory of staging area s, the object ID field with the object ID obtained in step 1625, and the branch field with the identification of the branch along which the edition lies.
>
> By taking the generation ID of the staging area, module 1316 essentially converts staging area s into the new edition. Consequently, when the edition is created, the edition creation module 1316 obtains 1640 a new generation ID and assigns 1650 it to the staging area.

> FIGS. 17a-17c illustrate the operation of the branch creation module 1346. To create a branch, the module 1346 determines 1705 whether the branch being created is a main branch. If the branch is not a main branch, the module 1346 obtains 1710 the generation ID of the base edition, call it "edition b," from the parent branch from which the new branch stems. Edition "b" can be any edition on the parent branch. The module 1346 then obtains 1715 from generation ID module 1348 a new generation ID derived from the generation ID of edition b. An initial edition, call it "edition e," is then created 1720 with the same root directory as edition b and with the generation ID obtained in step 1715. Additionally, zero or more work areas may be created 1730, each having the same root directory as edition e and a different generation ID derived from the generation ID of edition e. The work areas may be created at the time of branch creation or at a later time. A staging area is created 1740 having the same root directory as edition e and having a generation ID derived from the generation ID of edition e. Edition e, the work areas, and the staging areas are created in accordance with the methods discussed with respect to FIGS. 14-16.

This selection deals with (1) the ability to created multiple staging areas, (2) the operation of the edition creation module 1316 for creating an edition out of the contents of the staging area, and (3) the operation of the branch creation module 1346 which determines whether the branch being created is the main branch. None of these teachings, nor any other of the teachings from *Nakano* teaches storing a blank index file in the staging folder, wherein the blank index file inhibits default directory listing of the staging folder in the file system of the live server, as required by claim 5. *Nakano*, thus, does not teach this limitation of claim 5. Applicant, therefore, respectfully requests the Board to overrule the Examiner's rejection of claim 5.

G. Claim 7

Claim 7 requires, "... determining a length of said randomized string, wherein said length corresponds to a desired level of security." The Examiner contends that *Nakano* teaches this limitation at column 8, lines 52-62. Column 8, lines 52-62 was reproduced above for the convenience of the Board. As will be noted by the Board, there is no discussion in this selection with regard to selecting a length of the randomized string in order to have any type of affect on security. In fact, there is no disclosure in *Nakano* that discusses manipulating string length for security. Thus, *Nakano* does not teach the limitations of claim

7. Applicant, therefore, respectfully requests the Board to overrule the Examiner's rejection of claim 7.

H. Independent Claim 8

The Examiner rejected claim 8 stating, "Claim 8 has substantially the same limitations as claims 1 and 9." Final Office Action, p. 3. Claim 8 requires, "

> ... detecting, in a file system of a Web server, an index of said Web server;
>
> generating, in said file system of said Web server, a staging folder;
>
> storing, in said staging folder, said file assets to be staged according to names that include a random string, wherein said file assets are served by said Web server to users accessing said staging folder; and
>
> inhibiting listing of said staging folder in a default directory listing of said file system by said Web server.

The Web server referred to in claim 8 is also a "live" server. It contains the Web page files that are accessible by the public, hence, it is "live." As noted above, *Nakano* does not teach detecting an index file on the live server/Web server, generating a staging folder on the live server/Web server, storing file assets in the staging folder according to names including random strings, or inhibiting the listing of the staging folder in a default directory listing of the file system by the live server/Web server. Thus, *Nakano* does not teach any of the limitations of claim 8. Applicant, therefore, respectfully requests the Board to overrule the Examiner's rejection of claim 8.

I. Claim 9

The Examiner rejected claim 9 stating, "Claim 9 has substantially the same limitations as claim 3." Final Office Action, p. 3. Claim 9 requires:

> The computer program product of claim 8 wherein said detecting comprises:
>
> creating a temporary folder in root directory;
>
> writing a test file to said root directory using one of a plurality of standard index locations;

> requesting access to said test file from said Web server;
>
> writing said test file to said temporary folder using another of said plurality of standard index locations when said test file cannot be accessed during said requesting; and
>
> certifying said one of said plurality of standard index locations when said test file is accessed during said requesting.

As noted above with respect to claim 3, *Nakano* does not teach creating the temporary folder in the root directory, writing a test file to the root directory using one of the standard index locations, requesting access to the test file from the Web server, writing the test file to the root directory using another standard index location if the previous one did not work, and then certifying the correct index location when the requested access to the test file is successful. Thus, *Nakano* does not teach any of the limitations of claim 9. Applicant, therefore, respectfully requests the Board to overrule the Examiner's rejection of claim 9.

J. Claim 10

The Examiner rejected claim 10 stating, "Claim 10 has substantially the same limitations as claim 4." Final Office Action, p. 3. Claim 10 requires, "The computer program product of claim 9 wherein said plurality of standard addresses is stored on a memory accessible by a development environment." Again, as noted above, *Nakano* does not teach or even suggest the plurality of standard addresses, as required in claim 10. Thus, *Nakano* does not teach any of the limitations of claim 10. Applicant, therefore, respectfully requests the Board to overrule the Examiner's rejection of claim 10.

K. Claim 11

The Examiner rejected claim 11 stating, "Claim 11 has substantially the same limitations as claim 5." Final Office Action, p. 3. Claim 11 requires:

> The computer program product of claim 8 wherein said generating further comprises:
>
> creating an empty index named according to said detected index; and
>
> storing said empty index in said staging folder, wherein said empty index inhibits said listing of said staging folder in said default directory listing of said file system by said Web server.

As noted above, *Nakano* does not teach anything related to creating an empty index named according to the detected index and then storing the empty index in the staging folder to inhibit listing of the staging folder in default directory listings of the file system by the live server/Web server. Thus, *Nakano* does not teach any of the limitations of claim 11. Applicant, therefore, respectfully requests the Board to overrule the Examiner's rejection of claim 11.

L. Claim 13

The Examiner rejected claim 13 stating, "Claim 13 has substantially the same limitations as claim 7." Final Office Action, p. 4. Claim 13 requires, "... wherein a length of said random string is determined, wherein said length corresponds to a desired level of security." As noted above, *Nakano* does not even address security or measures to affect a level of security. Thus, *Nakano* does not teach any of the limitations of claim 13. Applicant, therefore, respectfully requests the Board to overrule the Examiner's rejection of claim 13.

M. Independent Claim 14

The Examiner rejected claim 14 stating, "Claim 14 has substantially the same limitations as claims 1 and 5." Final Office Action, p. 4. Claim 14 requires:

> A method for reviewing proposed file content on a live Web server comprising:
>
> scanning said live Web server for an index file;
>
> opening a review folder in a file system of said live Web server;
>
> creating a blank index on said review folder, wherein said blank index is named according to a name of said index file, wherein said blank index inhibits listing by said Web server of said review folder in a default directory listing of said file system; and
>
> storing said proposed file content in said review folder, wherein a randomized string is included in a file name representing said proposed file content, and wherein said proposed file content is served by said Web server to users accessing said file name.

As discussed at length above, *Nakano* does not teach scanning a live Web server for an index file, opening a review folder in a file system of the live Web server, creating a blank index on

the review folder, or storing a proposed file in the review folder using a randomized string in the file name. Moreover, *Nakano* does not teach that a review folder even exists on a live Web server, as required in claim 14. *Nakano* teaches two separate servers: a development server 130 and a website production server 170. Col. 5, lns 5-21. Thus, *Nakano* does not teach any of the limitations of claim 14. Applicant, therefore, respectfully requests the Board to overrule the Examiner's rejection of claim 14.

N. Claim 15

The Examiner rejected claim 15 stating, "Claim 15 has substantially the same limitations as claim 2." Final Office Action, p. 4. Claim 15 requires, "... sharing said file name with a reviewing user." As noted above, however, *Nakano* only teaches that developers can integrate their work by uploading it all to the staging area within which the developers may collaborate on further modifications. Col. 6, lns 26-36. This is not the same as sharing the file name, which has the randomized string inserted into it, with a reviewing user. Thus, *Nakano* does not teach any of the limitations of claim 15. Applicant, therefore, respectfully requests the Board to overrule the Examiner's rejection of claim 15.

O. Claim 16

The Examiner rejected claim 16 stating, "Claim 16 has substantially the same limitations as claim 3." Final Office Action, p. 4. Claim 16 requires:

> The method of claim 14 wherein said scanning comprises:
>
> creating a temporary folder on said web root of said Web server;
>
> retrieving a first index address from a plurality of standard index addresses;
>
> writing a test document to said temporary folder using said first index address;
>
> requesting said test document from said live Web server over hypertext transfer protocol (HTTP); and
>
> marking said first index address valid when said test document is retrieved during said requesting step.

Applicant reiterates that *Nakano* does not teach scanning the live Web server for an index file. Thus, this process for such a scanning, as described in claim 14, would never exist in

Nakano. As noted above, *Nakano* does not teach this "trial and error" method of trying to find the index file successively using multiple known standard index addresses. Thus, *Nakano* does not teach any of the limitations of claim 16. Applicant, therefore, respectfully requests the Board to overrule the Examiner's rejection of claim 16.

P. Claim 17

Claim 17 depends from claim 16 and provides detailed description of the scanning process when the test document cannot be retrieved in response to the requesting step of claim 16. Claim 17 requires, "... retrieving a next index address from said plurality of standard index addresses when said test document is not retrieved during said requesting step ...." The Examiner contends that *Nakano* teaches this limitation at column 9, lines 21-40. Final Office Action, p. 4. Lines 21-40 provide:

> In the directory 1200 illustrated in FIG. 12, the directory pages 1280 indicate that item "m" was added to the directory 1200 in generation G0, that item "n" was added to the directory 1200 in generation G2, and that "p" was added to the directory 1200 in generation G3. Note that, assuming nothing has been removed from the directory 1200, the directory pages 1280 include not only the contents that were added in a particular generation, but also the contents that were added in the generations from which the particular generation was derived. For instance, assume that the genealogy tree illustrated in FIG. 10 applies to this example. In generation G0, directory d1 contains item "m." In generation G2, item "n" is added to directory 1200, and, therefore, since G2 was derived from G0, directory 1200 contains items "m" and "n" in generation G2 (assuming "m" was not removed). In generation G3, item "p" is added to directory d1, and, therefore directory 1200 includes the items "p" and "m." Note that G3 was not derived from G2, and, consequently, directory 1200 does not include item "n" in generation G3.

This selection from *Nakano* deals with the directory pages within this directory 1200 that indicate the modification history of the directory. Neither this nor any other teaching from *Nakano* mentions retrieving a next index address from the plurality of standard index addresses when the test document is not retrieved during the requesting step, as required in claim 17. Thus, *Nakano* does not teach this limitation of claim 17.

Claim 17 further requires, "... writing said test document to said temporary folder using said next index address ...." The Examiner contends that *Nakano* teaches this limitation at the same column 9, lines 21-40 provided above. Final Office Action, p. 4. Clearly this cited selection has nothing to do with writing a test document to a temporary folder using the next standard index address. Neither this nor any other disclosure of *Nakano* teaches such a limitation. Thus, *Nakano* does not teach this limitation of claim 17.

Claim 17 further requires, "... requesting said next test document from said live Web server ...." Again, the Examiner contends that *Nakano* teaches this limitation at the same column 9, lines 21-40 provided above. Final Office Action, p. 4. However, the Board can clearly see that this selection does not discuss anything that could be analogized to a test document or requesting such a test document from the live Web server. Thus, *Nakano* does not teach this limitation of claim 17.

Claim 17 further requires, "... marking said next index address valid when said test document is retrieved during said requesting said next index step ...." Again, as before, the Examiner contends that *Nakano* teaches this limitation at the same column 9, lines 21-40 provided above. Final Office Action, p. 4. *Nakano* does not address validating index addresses when it is successfully able to retrieve a test document after a request, as required by claim 17. Thus, *Nakano* does not teach this limitation of claim 17.

Claim 17 further requires, "... repeating from said retrieving said next index step when said next test document is not retrieved during said requesting said next index step." As before, the Examiner contends that *Nakano* teaches this limitation at the same column 9, lines 21-40 provided above. Final Office Action, p. 4. Neither this selection nor any other disclosure from *Nakano* teach repeating the "trial and error" process to find the correct index address, as described in claim 17. Thus, *Nakano* does not teach any of the limitations of claim 17. Applicant, therefore, respectfully requests the Board to overrule the Examiner's rejection of claim 17.

Q. Claim 18

Claim 18 requires:

> ... determining a level of security for said proposed file content;
>
> establishing a number of characters for said randomized string responsive to said determined level of security; and
>
> generating a random character for each of said number of established characters.

The Examiner contends that *Nakano* teaches these limitations at column 8, lines 52-62. Lines 52-62 provide:

> FIG. 10 illustrates the method for deriving a generation ID, which is to be assigned to a new area, from a parent generation ID, where the parent generation ID is assigned to the direct parent of the new area. A unique number is obtained 1000 using a conventional algorithm for sequentially (e.g., 1, 2, 3, 4) or randomly generating unique numbers. The set or sequence of unique numbers associated with the parent generation ID is then retrieved 1010. Subsequently, a set or sequence of numbers that is the concatenation of the parent generation ID and the just issued unique number is created 1020.

As noted previously, this selection discusses one alternative in which a randomly generated unique number is used in creating a new generation ID from an old generation ID. Neither this nor any other disclosure from *Nakano* teaches or even discusses security or any means by which a level of security may be affected. Thus, *Nakano* does not teach any of the limitations of claim 18. Applicant, therefore, respectfully requests the Board to overrule the Examiner's rejection of claim 18.

R. Claim 20

Claim 20 requires, "... wherein said staging folder is not discoverable through a default directory listing of said file system by said server." The Examiner contends that *Nakano* teaches this limitation at column 6, lines 40-58 and column 7, lines 1-7. As referenced and commented on above, these cited selections from *Nakano* do not discuss anything about any folder, not just a staging folder, that is not discoverable through a default directory listing. Thus, *Nakano* does not teach any of the limitations of claim 20 Applicant, therefore, respectfully requests the Board to overrule the Examiner's rejection of claim 20.

S. Conclusion

In summary, the Federal Circuit requires that the ***identical invention*** be described in as great a detail as in the claims in order to support a rejection under 35 U.S.C. § 102. *Suzuki Motor Co.*, 868 F.2d at 1236. In the present examination, the asserted *Nakano* reference does not teach, disclose, discuss, or even suggest the clear majority of the limitations of claims 1-20. As such, Application respectfully requests the Board to overrule the Examiner's rejections of claims 1-20.

VIII. CLAIMS APPENDIX

A copy of the claims involved in the present appeal is attached hereto as Appendix A.

IX. EVIDENCE APPENDIX

No evidence pursuant to §§ 1.130, 1.131, or 1.132 or entered by or relied upon by the examiner is being submitted.

X. RELATED PROCEEDINGS APPENDIX

No related proceedings are referenced in II. above, hence copies of decisions in related proceedings are not provided.

Dated: July 31, 2008

Respectfully submitted,

By_____

Thomas J. Meaney
Registration No.: 41,990
FULBRIGHT & JAWORSKI L.L.P.
2200 Ross Avenue, Suite 2800
Dallas, Texas 75201-2784
(214) 855-8278
(Fax) (214) 855-8200
Attorney for Applicant

APPENDIX A

1. (Previously Presented) A method for staging file assets on a live server comprising:

detecting an index page of said server;

creating a staging folder within a file system of said server, wherein said staging folder does not default to a directory listing of said file system when accessed;

inserting a randomized string into a name of said file assets to be staged; and

storing said file assets in said staging folder.

2. (Original) The method of claim 1 further comprising:

communicating said name with said inserted randomized string to a reviewing party.

3. (Original) The method of claim 1 wherein said detecting step comprises:

creating a temporary folder to a root directory of said live server;

writing a page of content to said temporary folder according to one of a plurality of standard addresses;

attempting to access said temporary folder over hypertext transfer protocol (HTTP) from said server;

writing said page of content to said temporary folder using another of said plurality of standard addresses when said file cannot be read in said attempting step; and

validating said one of said plurality of standard addresses when said index page is read back in said attempting step.

4. (Previously Presented) The method of claim 3 wherein said plurality of standard addresses is stored on a memory accessible by a development environment.

5. (Previously Presented) The method of claim 1 wherein said creating step further comprises:

generating a blank index file named according to said detected index file; and

storing said blank index file in said staging folder, wherein said blank index file inhibits default directory listing of said staging folder in said file system.

6. (Original) The method of claim 1 further comprising:

generating said randomized string prior to said inserting step.

7. (Original) The method of claim 1 determining a length of said randomized string, wherein said length corresponds to a desired level of security.

8. (Previously Presented) A computer program product having a computer readable medium with computer program logic recorded thereon for facilitating staging file assets, wherein said computer program logic comprises code that when executed by a computer causes the computer to perform a method comprising:

detecting, in a file system of a Web server, an index of said Web server;

generating, in said file system of said Web server, a staging folder;

storing, in said staging folder, said file assets to be staged according to names that include a random string, wherein said file assets are served by said Web server to users accessing said staging folder; and

inhibiting listing of said staging folder in a default directory listing of said file system by said Web server.

9. (Previously Presented) The computer program product of claim 8 wherein said detecting comprises:

creating a temporary folder in root directory;

writing a test file to said root directory using one of a plurality of standard index locations;

requesting access to said test file from said Web server;

writing said test file to said temporary folder using another of said plurality of standard index locations when said test file cannot be accessed during said requesting; and

certifying said one of said plurality of standard index locations when said test file is accessed during said requesting.

10. (Previously Presented) The computer program product of claim 9 wherein said plurality of standard addresses is stored on a memory accessible by a development environment.

11. (Previously Presented) The computer program product of claim 8 wherein said generating further comprises:

creating an empty index named according to said detected index; and

storing said empty index in said staging folder, wherein said empty index inhibits said listing of said staging folder in said default directory listing of said file system by said Web server.

12. (Previously Presented) The computer program product of claim 8 further comprising:

generating said random string prior to said storing.

13. (Original) The computer program product of claim 8 wherein a length of said random string is determined, wherein said length corresponds to a desired level of security.

14. (Previously Presented) A method for reviewing proposed file content on a live Web server comprising:

scanning said live Web server for an index file;

opening a review folder in a file system of said live Web server;

creating a blank index on said review folder, wherein said blank index is named according to a name of said index file, wherein said blank index inhibits listing by said Web server of said review folder in a default directory listing of said file system; and

storing said proposed file content in said review folder, wherein a randomized string is included in a file name representing said proposed file content, and wherein said proposed file content is served by said Web server to users accessing said file name.

15. (Original) The method of claim 14 further comprising:

sharing said file name with a reviewing user.

16. (Original) The method of claim 14 wherein said scanning comprises:

creating a temporary folder on said web root of said Web server;

retrieving a first index address from a plurality of standard index addresses;

writing a test document to said temporary folder using said first index address;

requesting said test document from said live Web server over hypertext transfer protocol (HTTP); and

marking said first index address valid when said test document is retrieved during said requesting step.

17. (Original) The method of claim 16 further comprising:

retrieving a next index address from said plurality of standard index addresses when said test document is not retrieved during said requesting step;

writing said test document to said temporary folder using said next index address;

requesting said next test document from said live Web server;

marking said next index address valid when said test document is retrieved during said requesting said next index step; and

repeating from said retrieving said next index step when said next test document is not retrieved during said requesting said next index step.

18. (Original) The method of claim 14 further comprising:

determining a level of security for said proposed file content;

establishing a number of characters for said randomized string responsive to said determined level of security; and

generating a random character for each of said number of established characters.

19. (Previously Presented) The method of claim 1 further comprising:

said server serving said file assets to users who access said staging folder.

20. (Previously Presented) The method of claim 19 wherein said staging folder is not discoverable through a default directory listing of said file system by said server.

APPENDIX B

<<List/describe contents here>>

APPENDIX C

<<List/describe contents here.>>